PreFPIX2: Core Architecture and Results[1]

J. Hoff, A. Mekkaoui, D. Christian, S. Zimmerman, G. Cancelo, R. Yarema
Fermilab, P.O. Box 500, Batavia, Illinois 60510

Abstract

FPIX is a pixel architecture designed for colliding-beam experiments at the Tevatron. Its most important application to date is the BTeV experiment. PreFPIX2 is a chip designed to test the FPIX Core, i.e. the pixel control and readout architecture. This FPIX Core will be mated to a Periphery specific to a particular experiment.

Earlier plans called for the BTeV FPIX chip to be designed in a rad-hard process. However, deep-submicron CMOS processes have demonstrated appropriate radiation tolerance at a lower cost and with greater reliability. Therefore, PreFPIX2 has been fabricated in a 0.25 micron process utilizing radiation tolerant design techniques.

The architecture has undergone substantial development from earlier versions of FPIX. Most notable are the improvements to the column token passing scheme and to the End-of-column logic. Extensive simulations were performed using both SPICE and structural-level Verilog. Monte Carlo physics simulations of the BTeV pixel detector at half, full and double the planned luminosity were converted to Verilog compatible input files for the chip simulations, allowing the designers to observe the chip operating under real conditions and for extended periods of time. Analyses of the results reveal that at all luminosities the FPIX Core correctly identifies better than 99.6% of input hits. Bench tests of fabricated chips confirm the accuracy of the simulations.

I. INTRODUCTION

The FPIX architecture has been under development at Fermilab for the last three years. At present, the driving force behind this work is the BTeV experiment [1] though significant effort has been made in the design process to broaden the applicability of this architecture to fit any Tevatron experiment. In BTeV, the pixel detector will be employed for on-line track finding for the lowest level trigger system [2], and therefore, the pixel chips are required to read out all hit data from every crossing [3]. Given the expected distance from the beam (6mm for the nearest chips) and the expected luminosity ($2x10^{32}$ cm^{-2} s^{-2}), this means that the chip closest to the beam line will be expected to read out an average of 1.25 pixels per BCO crossing with statistical fluctuations often much higher.

PreFPIX2 is a developmental step in the evolution of the final FPIX architecture [4]. After the completion of FPIX1, it became apparent that certain functions such as pixel control and readout were independent of the requirements of any given experiment but that other functions, like data packing and communication with data acquisition hardware, changed for every experiment. Those functions independent of experimental requirements were organized into the FPIX Core. Experimentally dependent requirements were organized into the Periphery. PreFPIX2 represents the completion of the FPIX Core architecture. The FPIX Periphery specific to the BTeV experiment is under development.

The chip was developed in a 0.25μm process, using radiation tolerant design techniques such as enclosed transistors [5]. Recent tests show that this will enable FPIX to achieve the desired radiation tolerance (25-30 Mrad) without resorting to a rad-hard process [6]. PreFPIX2 has been developed to test a number of algorithmic and electrical modifications to the original read-out control system developed in FPIX1 [4]. Most notably, the programmable ability to operate in either an externally triggered or a self-triggered mode has been eliminated. The FPIX Core itself is now purely self-triggered.

In Section II, the FPIX Core architecture will be described. Section III will cover major developments to the FPIX architecture. Finally, Section IV will discuss the detailed simulation technique employed in preFPIX2's development and the results obtained thus far.

II. FPIX ARCHITECTURE

A. Core versus Periphery

The basic job of the FPIX Core is to convert the various pixel hits into a predictable data stream (see Figure 1). Whenever coreTalking is active, the Core will output a new data word (coreData) with every rising edge of the Readout clock. Moreover, coreData will be stable by the falling edge of the Readout clock . When coreTalking is inactive, coreData is zero.

CoreData itself is a 24-bit wide word that consists of three bits for hit magnitude from a 3-bit FADC located in each Pixel Cell, five bits for column location, eight bits for row location, and eight bits for time stamp.

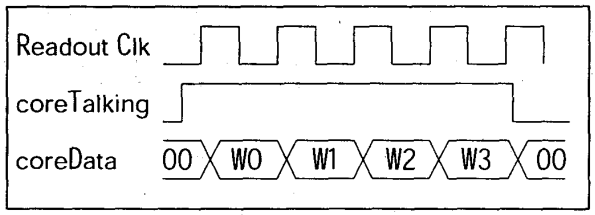

Figure 1 FPIX Core Data Stream.

[1] This work is sponsored by the U.S. Department of Energy under contract No. DE-AC02-76CH03000

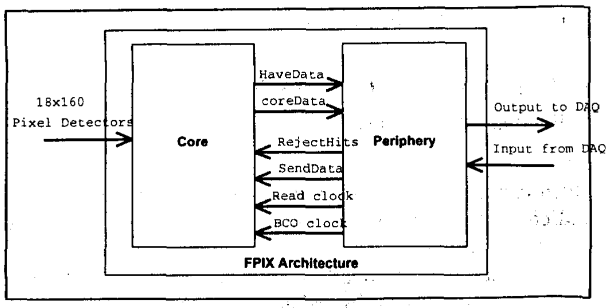

Figure 2 The Core - Periphery arrangement in the FPIX architecture

The Periphery communicates with the Core by providing the clocks and two control signals, RejectHits and SendData. When RejectHits is active, the Core is instructed to stop accepting new hits. Pixels already hit can still be read out. When SendData is active, the Periphery is telling the Core that it is ready to accept data. When SendData is inactive, the Core will send no more data, but it can continue to accept hits.

Using the structure in Figure 2, it is possible to imagine a wide range of Periphery cells customized to an experiment's needs.

B. Core Organization

The FPIX Core is a column-based architecture that uses an indirect addressing scheme to associate pixel hits with a time stamp. It is best understood as consisting of three mutually dependent functional blocks as shown in Figure 3. These three blocks are the Core Logic, the End-of-column Logic and the Pixel Cell.

The Core Logic understands time. At the rising edge of the beam clock, it stamps every time slice with an eight-bit beam crossing (BCO) number. This number is broadcast to all End-of-column Logic blocks. The Core Logic also contains a very simple state machine that knows if the Core is Talking or Silent. The Core is Silent until the End-of-column Logic blocks indicate that they have data to output. When this happens, at the next rising edge of the Readout

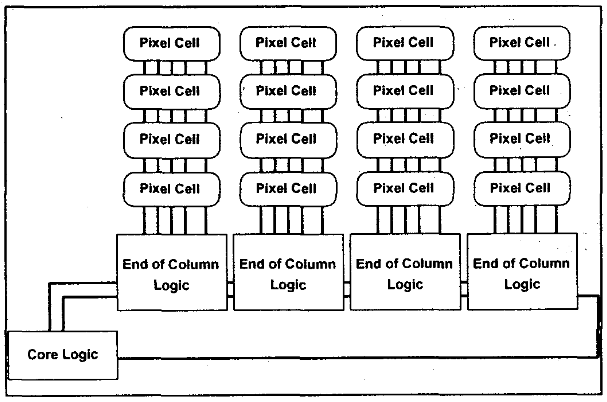

Figure 3 The FPIX Core Organization. In a full FPIX chip, there would be 18 columns with 160 Pixel Cells per column

clock, the Core will switch to the Talking state, and output can begin. The Core Logic does not wait for any kind of chip token or validation from the Data Acquisition (DAQ) hardware. That would be the job of the Periphery block. While in the Talking state, the Core starts passing a Horizontal Token across the End-of-column Logic blocks to arbitrate rights to the output bus. When the Horizontal Token drops out of the other side of the End-of-column Logic blocks readout is done. The Core switches back to the Silent state at the next rising edge of the Readout clock.

The End-of-column Logic blocks are considerably more complicated. They also understand time in that, whenever there is a hit, they store the BCO numbers broadcast by the Core Logic. Obviously, they also understand hits, which are driven to them from the Pixel Cells via the HFastOR signal, a distributed OR-gate with a pull-down transistor in every Pixel Cell in the column. The End-of-column Logic Blocks also understand the existence of the Pixel Cells because they communicate to those pixels through a series of commands and tokens. Finally, the End-of-column Logic Blocks understand output. When the Core is in the Talking state and when a particular End-of-column Logic Block has the Horizontal Token and when that End-of-column Logic Block has hit pixels to output, then it outputs those pixels.

The Pixel Cells themselves know nothing of time. They only understand hits and commands from their End-of-column Logic block. These commands are Idle (do nothing), Listen (listen for new hits), Reset (reset your contents), and Output (output your contents). There are four sets of such commands coming from the End-of-column Logic block. If a pixel is Empty and it receives a hit, it associates itself with whichever command set is issuing the Listen command. From that point and until the Pixel Cell is reset, it obeys only the commands from the associated command set. The Pixel Cells also communicate back to the End-of-column Logic bock via the fast ORs. The HFastOR communicates hits in response to a Listen command and the RFastOR communicates the presence of Pixel Cells as yet unread in response to an Output command. Rights to the column output bus are arbitrated by a Column Token issued by the End-of-column Logic block.

The original FPIX architecture included the ability to switch between an externally triggered or self-triggered mode. In the externally triggered mode, an external source provided the chip with a BCO number which was compared to the BCO numbers latched in the End-of-column Logic Blocks. In the self-triggered mode, a second BCO counter broadcast requested BCO numbers. If there was a match with any stored BCO number in any End-of-column Logic block, then the counter would be stopped and all hit pixels associated with that BCO number would be read out. This constant need to compare requested BCO numbers to stored BCO numbers reduced the efficiency of the readout scheme. In preFPIX2, there are no such BCO comparisons. Instead, if any End-of-column Logic block has any data to output, it immediately alerts the Core Logic, which then switches to the Talking state. Unlike the original output scheme, this method does not guarantee that hit pixels would be output in time stamp order. However, the new output scheme dramatically improves readout efficiency.

III. Developments in PreFPIX2

A. End-of-column Logic

The original FPIX architecture utilized four Command State Machines, one for each command set. The state machines were simple, with only two states, Empty and Full. However, since the state machines made their transitions at the rising edge of the BCO clock, great pains were necessary to ensure that information synchronous to the Readout Clock, such as the completion of an output, arrived to these state machines with enough setup and hold time. Moreover, the original architecture required a priority encoder state machine to determine which command set would be the next to issue the Listen Command. This required a substantial amount of room, and created some timing problems of its own.

In the new FPIX Core, these problems are solved as shown in Figure 4. First, there is the addition of the Column State Machine, which operates at the rising edge of the Readout Clock. Second, the simple two-state Command State Machines are replaced with four-state Command State Machines that still operate on the rising edge of the BCO clock. By developing the state machines with different clocks, the synchronicity problems are eliminated. The Column State Machine governs all activity that must be synchronous with the Readout Clock. The Command State Machines govern all activity that must be synchronous with the BCO Clock.

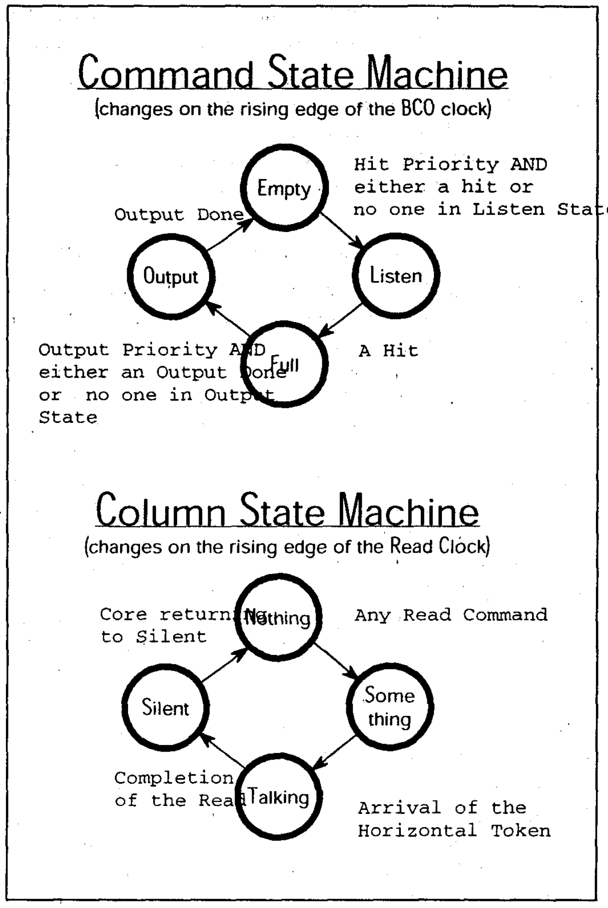
Command State Machine
(changes on the rising edge of the BCO clock)
Empty
Listen
Full
Output
Hit Priority AND either a hit or no one in Listen State
A Hit
Output Priority AND either an Output Done or no one in Output State
Output Done
Column State Machine
(changes on the rising edge of the Read Clock)
Nothing
Something
Talking
Silent
Any Read Command
Arrival of the Horizontal Token
Completion of the Read
Core returns to Silent

Figure 4 End-of-column Logic State Machines

A simple, purely combinatorial priority encoder chooses which Command State Machine will be the next to enter the Listen state from among those state machines currently in the Empty State. In the Empty State, the Command State Machines issue the Idle Command. In the Listen State, they issue the Listen Command. Once a hit is received, several things happen. First, the BCO number currently being broadcast by the Core Logic is stored in a register associated with the Command State Machine currently in the Listen State. Second, that state machine makes the transition to the Full State where it once again issues the Idle command. Third, the state machine picked by the Hit Priority Encoder as next to Listen moves to the Listen State.

Since it is possible for more than one Command State Machine to be in the Full state at the same time, a second priority encoder is required. This Output Priority Encoder is necessarily more complicated than the Hit Priority Encoder. For example, if the DAQ system can read a chip faster than hits are input to it, then low priority Command State Machines may never enter the Listen state, and this would have no effect on our efficiency. High priority Command State Machines would do all the work. However, if a state machine enters the Full state, then it must reach the Output state as quickly as possible or that data will be lost. In other words, somehow all machines in the Full state must have equal priority while, at the same time, something must distinguish them so that a choice can be made. Finally, to minimize the transistor count and to maintain the isolation of the Readout and BCO clocks, the Output Priority Encoder must also be purely combinatorial. The solution is to rely on the states of the Command State Machines and to use a circular scheme as shown in Figure 5. If State Machine A is in the Output state, then State Machine B has the highest priority in the Output Priority Encoder, then C and then D. If State Machine B is in the Output state, then State Machine C has the highest priority then D and then A. If no one is in the Output state, then State Machine A has the highest priority. Once in the Output State, the state machine issues the Output Command until it receives the Output Done signal from the Column State Machine. At that point, the Reset Command is issued as a redundant means of making sure all pixels associated with this Command State machine are cleared.

The Column State Machine starts in the "Nothing to Say" or Nothing State where it remains until it sees an Output (Read) command issued by any of the Command State Machines. At the next rising edge of the Read Clock, the Column State Machine makes the transition to the "Something to Say" or Something State. At this point, the Core logic is alerted to the fact that there is data to output,

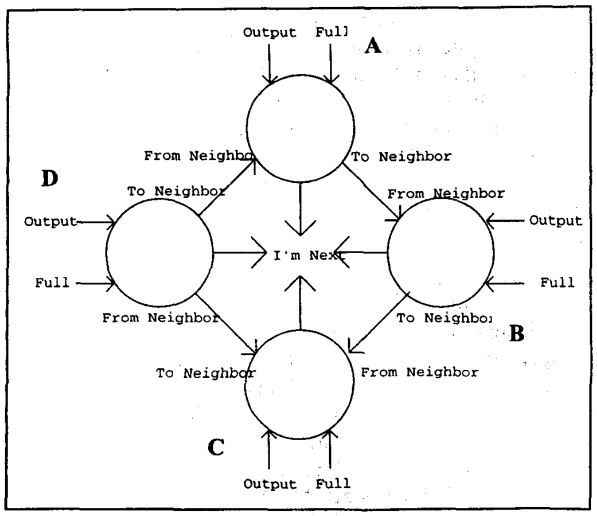

Figure 5 A Logical Diagram of the Output Priority Encoder

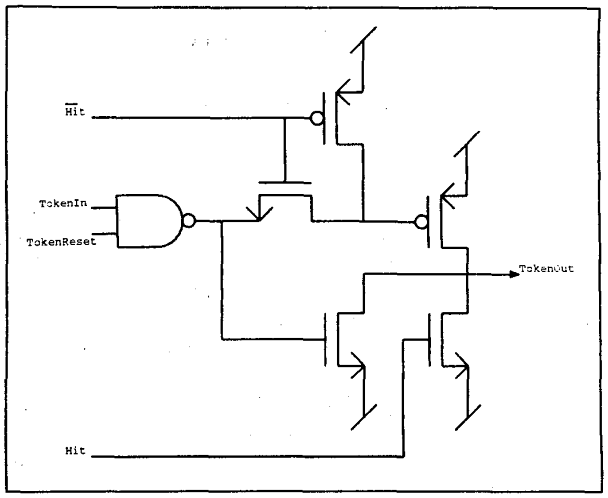

Figure 6: The Token Passing circuitry

and the Column Tokens are sent up to the first Pixel Cell that needs to be output. The state machine then waits for the arrival of the Horizontal Token from the Core Logic. When the token arrives, the Column State Machine makes the transition to the Talking state, and it releases the Readout clock to the Pixel Cells enabling them to output their data. Simultaneously, the stored BCO number (which associates the hit pixels with the time they were hit) is driven onto the bus. The last pixel is being read out when the RFastOR goes away. This signals the completion of the read cycle. At the next rising edge of the Read Clock, the Column State Machine makes the transition to the Silent state. This sets the Output Done signal informing the Command State Machine that it can make its own transition from the Output to the Empty State. The Output Done signal is reset when the Command State Machine reaches the Empty state. When the entire array has been read out, the Horizontal Token drops out of the last End-of-column Logic Block, and the Core Logic makes its transition from Talking to Silent. This signals to all the Column State Machines that they can make their own transition back to the Nothing State.

B. Token Passing Logic

Experience with earlier versions of FPIX has revealed that there are two limiting factors in readout speed related to the Column Token.

First, speed is limited by how fast the token can be passed from one hit pixel to another. Once a hit pixel has been ordered to Output, it waits for the Column Token, grabs it and then drives its data onto the bus at the rising edge of the Read Clock. Simultaneously, it releases the token to the next hit pixel in the column. Under worst case conditions, that token must travel almost the entire length of the column before it reaches the next hit pixel and it must do this before the next rising edge of the Read Clock. The amount of time it takes for the token to pass through an empty pixel is called the skip frequency. Therefore, the maximum readout speed is equal to the skip frequency divided by the number of pixels in the column.

Second, speed is limited by how long it takes an entire column to restore itself after a read has been completed. This determines how rapidly successive read cycles can be made.

The token passing architecture shown in Figure 6 has been optimized in preFPIX2 to permit skipping frequencies between 7 and 8GHz, yielding readout clock frequencies in excess of 40 MHz. It is also resettable, allowing for maximum speed in successive read cycles.

IV. MONTE CARLO-VERILOG SIMULATION

A unique and very comprehensive method of design and simulation was used on the FPIX Core. First, individual digital subcircuits including nand gates, nor gates, inverters, CMOS transmission gates, SR-flip-flops and D-flip-flops were simulated using SPICE to determine their best, worst and typical propagation delays. Next, all critical drivers such as Command Drivers and Address Drivers were similarly evaluated under their expected loads. All of these delays were transferred into the Verilog hardware description language. Then the readout architecture and control system were constructed in a bottom-up fashion from those basic digital components. No behavioral modeling was permitted in the FPIX Core, and great attention was given to ensuring that gates were not excessively loaded with capacitance. The net result was a Verilog model of the FPIX Core accurate to the gate level and, in many places, the transistor level. It modeled the entire data path from the output of each of the 2880 analog front-ends to the pads of the chip.

This procedure yielded a number of benefits. First, through software, purely structural Verilog code can be converted into schematics. Therefore, layout-versus-schematic (LVS) comparisons became, in effect, layout-versus-Verilog comparisons. Second, this design procedure ultimately produced a Verilog model of the Core that was extremely accurate with respect to timing. SPICE simulations were performed regularly at higher and higher levels of the hierarchy to ensure this.

Next, the analog front end of each pixel was modeled behaviorally. The model describes both the time walk and the dead time of the analog front end. Moreover, the model includes the way time walk and dead time change as a function of charge magnitude. The information necessary for this model was determined experimentally from prototype versions of the front end.

Monte Carlo analysis of 5000 beam crossover periods in the interaction chamber was done using MCFAST, making geometric cuts around the region that would be occupied by the hottest chip. This produced a list of hit pixels with their associated charge magnitudes in each of the 5000 time slices. Different analyses were made assuming the expected luminosity of the beam, half that luminosity and twice that luminosity. The results of these analyses were converted into Verilog and used as input to the FPIX Core model.

Finally, a rudimentary DAQ system was modeled to capture the output of the FPIX Core. This output was reconstructed into hit pixels, their hit magnitude and time stamp. The input and output lists were compared and additional lists were made of matches, missing members of the input list, and extra members of the output list (see Table 1).

Table 1
Simulated Results of the FPIX Core displayed by luminosity

Luminosity	# of hit pixels in the input list	# of unmatched members of the input list	# of unmatched members of the output list
$1x10^{32}$ cm^{-2} s^{-2}	1342	0	0
$2x10^{32}$ cm^{-2} s^{-2}	2751	2	3
$4x10^{32}$ cm^{-2} s^{-2}	11643	33	16

The majority of the unmatched members of the output list failed to match members of the input list due to an improper hit magnitude. This occurs when a single pixel is hit twice before the first hit can be read out. This happens because the behavioral model of the analog front end attempts to model reality. That is, the digital back end of the pixel would ignore the new hit because it was already full. However, the ADC in the analog front end would respond to the new hit by updating its value if the new hit was larger than the original hit. The net result in the output list is an "original hit" that has a correct time stamp, a correct address and a potentially incorrect magnitude. The new hit will be missing from the output list. The Verilog model of the analog front end does not care if the new hit is larger or smaller than the old hit. Therefore, the number of unmatched members of the output list shown in Table 1 is necessarily larger than expected in reality.

The number of unmatched members of the input list in Table 1 is usually related to situations where hits cannot be accepted by a pixel. For example, if all of the Command State Machines are either in the Full or Output states, then no Command State Machines are available to be transferred to the Listen state. Under these conditions, the column can accept no new hits, and therefore, those hits are missing from the output list.

V. Conclusions

The FPIX Core architecture has been completed. Substantial improvements were made to its architecture, resulting in readout efficiencies greater than 99.6%. Rigid adherence to bottom-up design techniques, with great attention paid at the start of the project to propagation delays in low-level digital cells, resulted in a Verilog model of the architecture that was accurate at the gate level to the final design. Therefore, the Verilog model could be compared to the final layout using standard CAD software. The model's timing was also very accurate even at the highest levels. Monte Carlo simulations of the interaction region performed by the physicists on the BTeV project were used as inputs to the model of the FPIX Core. This enabled the designers to exhaustively test the design. It also permitted the chip designers to present to the system designers a description of expected data stream coming from the chip.

VI. Acknowledgments

The authors would like to thank Al Dyer for his exceptional wire-bonding abilities and Gerry Dychakowsky for PC board layout. We would also like to thank Penny Kasper for the MCFAST Monte Carlo data.

VII. References

[1] A. Kulyatsev, *et al* "A Proposal for an Experiment to Measure Mixing, CP Violation and Rare Decays in Charm and Beauty Particle Decays at the Fermilab Collider – BTEV", Fermilab-Proposal-897.

[2] G. Cancelo, *et al*, "High Readout Speed Chip Developed at Fermilab", Fermilab-Conf-98/278, published in the Proceedings of the Fourth Workshop on Electronics for LHC Experiments, Rome, Italy, September, 1998

[3] D. Christian, "Development of a pixel readout chip for BTeV", Nuclear Instruments & Methods in Physics Research A, 435m 144-152 (1999).

[4] J. Hoff and A. Mekkaoui, 'FPIX Core Architecture and the PreFPIX2 Chip", Fermilab-TM-2110,

[5] W. Snoeys, "Radiation tolerance beyond 10 Mrad for a pixel readout chip in standard submicron CMOS", Proc. Of the 4th Workshop on Electronics for LHC Experiments, Rome, September, 1998, CERN/LHCC/98-36, 114

[6] A. Mekkaoui and J. Hoff, '30 MRad (SiO_2) Radiation tolerant pixel front end for the BTeV experiment" Proc. Of Pixel 2000, Genova, June, 2000.